Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №1

із дисципліни «Технології розроблення програмного забезпечення»
Тема: «Системи контроля версій. Розподілена система контролю версій «Git»**»**

Виконала:
студентка групи ІА-31:
Мартишенко І.С.

Перевірив:
Мягкий М.Ю.

Київ- 2025

Тема: Системи контроля версій. Розподілена система контролю версій «Git».

Мета: Навчитися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git.

1. Створити локальний репозиторій в пустій дирпекторії:

```
irynamartyshenko@Irynas-MacBook-Air ~ % git init lab01
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/irynamartyshenko/lab01/.git/
```

2. Додати довільний файл з довільним текстом та зафіксувати додавання файлу:

```
irynamartyshenko@Irynas-MacBook-Air lab01 % echo 'txt' > test.txt
irynamartyshenko@Irynas-MacBook-Air lab01 % git add .
irynamartyshenko@Irynas-MacBook-Air lab01 % git commit -m 'first commit'
[master (root-commit) 0eeaa8d] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt
```

3. Створити 3 нові бранчі(3 різними способами):

```
irynamartyshenko@Irynas-MacBook-Air lab01 % git branch branch_1
irynamartyshenko@Irynas-MacBook-Air lab01 % git checkout -b branch_2
Switched to a new branch 'branch_2'
irynamartyshenko@Irynas-MacBook-Air lab01 % git checkout master
Switched to branch 'master'
irynamartyshenko@Irynas-MacBook-Air lab01 % git switch -c branch_3
Switched to a new branch 'branch_3'
```

4. Створити 2 файли у бранчі master:

```
irynamartyshenko@Irynas-MacBook-Air lab01 % echo '1' > f1.txt
irynamartyshenko@Irynas-MacBook-Air lab01 % echo '2' > f2.txt
```

5. Зробити видимими ці 2 файли, але не комітити:

```
irynamartyshenko@Irynas-MacBook-Air lab01 % git add .
```

6. Перейти на branch_1 і зробити коміт файлу f1.txt:

```
    first commit
irynamartyshenko@Irynas-MacBook-Air lab01 % git checkout branch_1
A       f1.txt
A       f2.txt
Switched to branch 'branch_1'
irynamartyshenko@Irynas-MacBook-Air lab01 % git commit -m 'add f1' f1.txt
[branch_1 151f194] add f1
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt
```

7. Перейти на branch_2 і зробити коміт файлу f2.txt:

```
irynamartyshenko@Irynas-MacBook-Air lab01 % git checkout branch_2
A       f2.txt
Switched to branch 'branch_2'
irynamartyshenko@Irynas-MacBook-Air lab01 % git commit -m 'add f2' f2.txt
[branch_2 41a5317] add f2
 1 file changed, 1 insertion(+)
 create mode 100644 f2.txt
```

8. На branch_2 створити файл f1 з текстом b:

```
nothing to commit, working tree clean
irynamartyshenko@Irynas-MacBook-Air lab01 % echo 'b' > f1.txt
```

9. Об’єднати branch_2 з branch_3:

```
irynamartyshenko@Irynas-MacBook-Air lab01 % git switch branch_3
Switched to branch 'branch_3'
```

```
irynamartyshenko@Irynas-MacBook-Air lab01 % git merge branch_2
Updating 0eeaa8d..41a5317
Fast-forward
 f2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 f2.txt
```

10. Переписати історію поточної гілки поверх branch_1:

```
irynamartyshenko@Irynas-MacBook-Air lab01 % git rebase branch_1
Auto-merging f1.txt
CONFLICT (add/add): Merge conflict in f1.txt
error: could not apply 4077ea8... Add f1.txt
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
hint: Disable this message with "git config set advice.mergeConflict false"
Could not apply 4077ea8... Add f1.txt
```

11.Вирішити конфлікт (вибрати вхідні зміни):

```
irynamartyshenko@Irynas-MacBook-Air lab01 % code f1.txt
```

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
1
=======
b
>>>>>>> 4077ea8 (Add f1.txt) (Incoming Change)
```

12.Позначити конфлікт як вирішений:

```
irynamartyshenko@Irynas-MacBook-Air lab01 % git add f1.txt
```

Висновок:

У ході виконання роботи я ознайомилась з принципами роботи розподілених систем контролю версій на прикладі Git. Я навчилася виконувати основні операції - створення репозиторію, додавання файлів, фіксацію змін (commit), перегляд історії та роботу з віддаленим репозиторієм.

Відповіді на контрольні питання:

1. Що таке система контролю версій (СКВ)?

Система контролю версій - це програмний інструмент, який відстежує зміни у файлах проєкту, дозволяє повертатися до попередніх версій, працювати над різними гілками розвитку коду та координувати роботу кількох розробників.

2. Відмінності між розподіленою та централізованою СКВ

Централізована система зберігає всі дані на одному сервері, тому робота без мережі неможлива, а швидкість залежить від підключення. Розподілена система має повну копію репозиторію на кожному комп'ютері, дозволяє працювати офлайн і є більш надійною, оскільки дані дублюються у всіх копіях.

3. Різниця між stage та commit в Git

Stage - це підготовка змін до коміту. Commit - це збереження змін у локальному репозиторії. Отже, stage - це етап підготовки, а commit - фіксація.

4. Як створити гілку в Git

Для створення гілки використовується команда git branch <назва_гілки>. Щоб створити і відразу перейти в неї, використовують git checkout -b <назва_гілки>. Також новий спосіб створити гілку використовуючи git switch -c <назва>.

5. Як створити або скопіювати репозиторій Git з віддаленого серверу

Для копіювання репозиторію використовують команду git clone <url>, наприклад: git clone https://github.com/user/project.git. Це створює локальну копію проєкту.

6. Що таке конфлікт злиття, як створити і вирішити

Конфлікт злиття виникає, коли дві гілки змінюють один і той самий рядок по-різному. Щоб створити конфлікт, потрібно змінити один і той самий фрагмент у двох гілках і виконати git merge. Для вирішення конфлікту відкривають файл, вибирають правильний варіант, додають файл git add <файл> і завершують коміт командою git commit.

7. Команди merge, rebase, cherry-pick - коли використовуються

Merge - об'єднує зміни з іншої гілки, зберігаючи історію. Rebase - переносить коміти поверх іншої гілки, роблячи історію чистішою. Cherry-pick - копіює окремий коміт з іншої гілки.

8. Як переглянути історію змін Git репозиторію

Для перегляду історії використовують команду git log. Щоб побачити скорочений вигляд, застосовують git log --oneline.

9. Як створити гілку не використовуючи git branch

Гілку можна створити за допомогою команди git checkout -b <назва_гілки> або git switch -c <назва_гілки>.

10.Як підготувати всі зміни в поточній папці до коміту

Для цього використовують команду git add .

11.Як підготувати всі зміни в дочірній папці до коміту

Потрібно виконати git add <шлях_до_папки>, наприклад git add src/.

12.Як переглянути перелік наявних гілок в репозиторії

Для перегляду локальних гілок використовують git branch, а для перегляду всіх (у тому числі віддалених) - git branch -a.

13. Як видалити гілку

Локальну гілку видаляють командою git branch -d <назва_гілки>, а віддалену — git push origin --delete <назва_гілки>.

14. Способи створення гілки та в чому між ними різниця

Гілку можна створити трьома способами: git branch <назва> - створює гілку без переходу; git checkout -b <назва> - створює і одразу переходить у неї; git switch -c <назва> - новий, більш зручний спосіб створення гілки.